EECS 214 Spring 2016 Worksheet 1

Stacks

In a sentence, describe what a stack does at a high level. Namely, how does a stack differ from a queue?

For each of these stack operations, explain what they do and give the worst-case O(n) complexity.

- **Push**
- **Pop**
- **Peek**

For the rest of the section, assume the following class definition for a stack implemented using an array:

```
public class Stack
    {
        // Define object properties here
        object[] stack = new object[10];
        int first = 0;
        int last = 0;
        int size = 0;

        public override void Push(object o)
        { ... }

        public override object Pop()
        { ... }

        public override int Count
        { ... }

        public override object Peek()
```

```
        { ... }

        public override bool IsFull
        { ... }
    }
```

Assume you have initialized the following stack, `MyStack` :

```
Stack myStack = new Stack();
myStack.Push(3);
myStack.Push(1);
myStack.Push(-5);
myStack.Push(0);
myStack.Push(2);
```

What is the expected return value of each of the following operations? (Note that some operations will not have a return value; in this case, just write `void`)

```
myStack.Pop();    // Answer:
myStack.Push(4);  // Answer:
myStack.IsFull;   // Answer:
myStack.Pop();    // Answer:
myStack.Pop();    // Answer:
myStack.Peek();   // Answer:
myStack.Count;    // Answer:
myStack.Pop();    // Answer:
```

The following exercise will prepare you to write unit tests for a stack implementation.

For each stack operation, your goal is to comprehensively describe the operation's expected behavior in terms of example inputs and expected outputs. Then, use `Assert` to write each example as a valid unit test. Here is an example:

```
[TestMethod]
public void PushTest() {
  // Set up any structures you need for the test
  Stack myStack = new Stack();
  myStack.Push(4);
```

```
    myStack.Push(3);
    // Use assertions to test expected behaviors
    Assert.AreEqual(3, myStack.Pop());
}
```

You may use the initial definition of `MyStack` (before the operations in the previous question), and you may also define new `Stacks` for testing. Be sure to write enough examples to cover edge cases comprehensively (for example, consider how an operation should behave when a `Stack` is full).

Push
Should add the new value to the top of the stack

Should throw an exception if stack is full

Pop
Should remove the topmost value and return it

Should throw an exception if stack is empty

Count
Should return the correct size for a nonempty stack

Should return 0 for an empty stack

IsFull
Should return `false` when stack is empty and not full

Should return `true` when stack is full

Queues

In a sentence, describe what a Queue does at a high level

For each of the following Queue operations explain what they do and give the worst-case O(n) complexity.

- **Insert/Enqueue**
- **Remove/Dequeue**

For the following questions assume the following implementation of a Queue

```
1
2  public class Queue
3      {
4          // Define object properties here
5          object[] queue= new object[10];
6          int first = 0;
7          int last = 0;
8          int size = 0;
9
10         public override void Enqueue(object o)
11         { ... }
12
13         public override object Dequeue()
14         { ... }
15
16         public override int Count
17         { ... }
18
19         public override bool IsFull
```

```
        { ... }
        //checks to see if the Queue does not have more room
    }
```

Assume that we have created a Queue, MyQueue, via the following operations

```
Queue MyQueue = new Queue ();
//created our Queue
MyQueue.Enqueue(10);
MyQueue.Enqueue(20);
MyQueue.Enqueue(30);
MyQueue.Enqueue(40);
MyQueue.Enqueue(50);
```

What is the expected return value of the following operations? (similar to the Stack question, you may write void for a void return, however note that these operations still affect our queue)

```
MyQueue.Enqueue(100); //
MyQueue.Count(); //
MyQueue.Dequeue(); //
MyQueue.Enqueue(25); //
MyQueue.Dequeue(); //
MyQueue.IsFull(); //
```

Similar to the Stack test cases, write test cases that would ensure that our Queue implementation is working properly. You can use MyQueue from above (before we changed it).